ILab—Parameter Study Tool and IPG Portal

Maurice Yarrow

The ILab tool was specifically written to solve the complex and difficult problems of creating and launching parameter studies. Though today's distributed computational resources are quite capable of running large parameter studies for aerospace problems, users have not had tools available to them that make this process easy and fast. ILab was developed with a host of user-friendly features, so that creating and launching parameter studies can now be accomplished simply and in just minutes.

Within the last year, the ILab tool was given an intuitive user interface. Users can now specify their processes via an advanced computer-aided design (CAD) approach by visually constructing a flowchart-like graphic. ILab's code generator subsequently translates this into appropriate shell scripts, which are then launched onto remote systems and monitored.

Researchers whose parameter studies consist of individual jobs with long-running computational fluid dynamics problems can now take advantage of ILab's unique "restart" capability. This allows users to automatically segment their jobs onto supercomputer scheduling systems and to modify solver parameters for the purpose of steering the computation to a stable solution. Load balancing of jobs is automatically accomplished by restarting the jobs onto supercomputer systems with immediate processing availability. In addition, an advanced Help system has been built into ILab, which users can access from any ILab screen.

Pictured (fig. 1) is the special-purpose parameterization screen (bottom left) which automates the construction of input files for multi-dimension parametric studies. The monitoring screen (upper left) shows the progress of

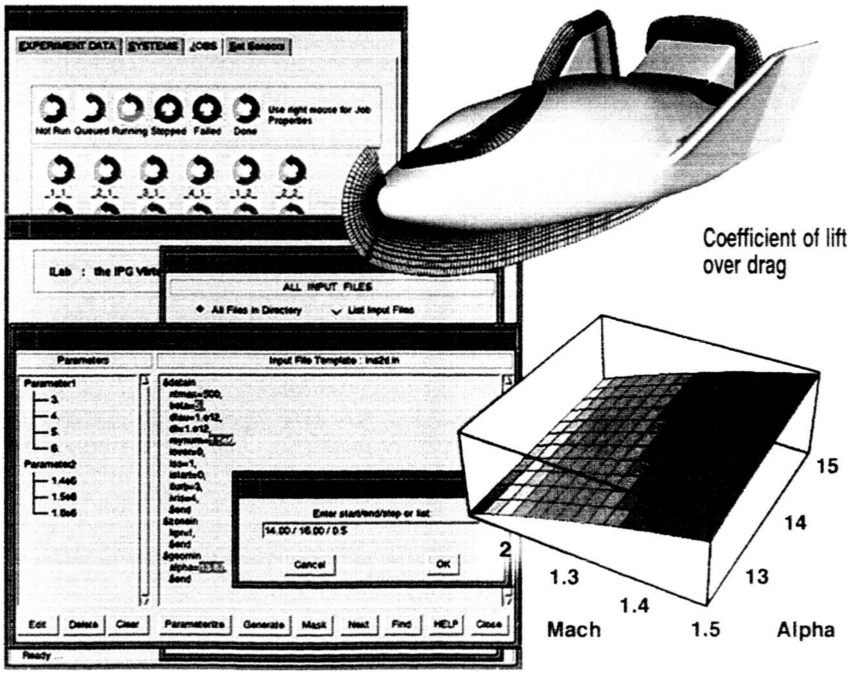

Fig. 1. The ILab Parameter Study Tool and results for the X-38 crew return vehicle.

individual jobs in a parameter study experiment. The NASA X-38 Crew Return Vehicle (upper right) was the subject of a two-dimensional parameter study in Mach number and angle of attack. ILab generated and submitted 192 separate flow-field computations for the requested 16 values of Mach number and 12 values of angle of attack. Pictured (bottom right) is the surface of coefficients of lift-over-drag for the X-38 at these 192 parameter combinations.

Point of Contact: Maurice Yarrow
(650) 604-5708
yarrow@nas.nasa.gov

Multithreading for Dynamic Unstructured Grid Applications

Rupak Biswas

The success of parallel computing in solving realistic computational applications relies on their efficient mapping and execution on large-scale multiprocessor architectures. When the algorithms and data structures corresponding to these problems are unstructured or dynamic in nature, efficient implementation on parallel machines offers considerable challenges. Unstructured applications are characterized by irregular data-access patterns whereas dynamic mesh adaptation causes computational workloads to grow or shrink at run time. For such applications, dynamic load balancing is required in order to achieve algorithmic scaling on parallel machines. Our objectives were to implement various parallel versions of a dynamic unstructured algorithm and to critically compare their performances in terms of run time, scalability, programmability, portability, and memory overhead.

A multithreaded version of a dynamic unstructured mesh-adaptation algorithm has been implemented on the Cray (formerly Tera) Multithreaded Architecture (MTA). Multithreaded machines can tolerate memory latency and utilize substantially more of their computing power by processing several threads of computation. For example, the MTA processors each have hardware support for up to 128 threads, and are therefore especially well suited for irregular and dynamic applications. Unlike traditional parallel machines, the MTA has a large uniform shared memory, no data cache, and is insensitive to data placement. Parallel programmability is significantly simplified since users have a global view of the memory, and need not be concerned with partitioning and load-balancing issues. Performance was compared with an MPI implementation on the T3E and the Origin2000, and a shared-memory directives-based implementation on the Origin2000.

A standard computational mesh simulating flow over an airfoil was used for our experiments to compare the three parallel architectures. The initial mesh, consisting of more than 28,000 triangles, was refined a total of five times to generate a mesh 45 times larger, as shown in figure 1. Performance by platform and programming paradigm is presented in figure 2. It is important to note that different parallel versions use different dynamic load-balancing strategies. The multithreaded implementation of the adaptation algorithm required adding a trivial amount of code to the original serial version and had little memory overhead. In contrast, the MPI version doubled the size of the code and required significant additional memory for the communication buffers. The simulation on the eight-processor MTA at San Diego Supercomputing Center using 100 threads per processor was almost ten